Matrix and Tensor Factorization for Machine Learning

African Master of Machine Intelligence
AIMS/AMMI

Instructor: *Guillaume Rabusseau*
DIRO & Mila - Université de Montréal - CIFAR AI Chair
email: grabus@iro.umontreal.ca

Course Overview

- Class overview:
  - Quick refresher on matrix decomposition techniques (eigendecomposition, SVD, ...)
  - Quick overview of matrix factorisation techniques in ML (PCA, completion, spectral graph clustering, ...)
  - Introduction to tensor decomposition techniques (CP, Tucker, Tensor Train decomposition)
  - Introduction to tensor networks.
  - Optimization techniques for tensor problems (gradient descent, alternating minimization)
- The lectures will be based on a course I give at Montreal University. Lots of material and references available on the class webpage.

Class organization

- Mornings (Mon. Tu. and Wed.): pre-recorded videos
- Afternoons: Discussion, Q&A and programming labs

Programming labs (organized by my student Tianyu Li):

Lab 1 Introduction to operations on tensors with einsum and the tensorly package

Lab 2 Image completion using gradient descent and tensor decomposition techniques

Lab 3 Compressing neural networks using tensor decomposition

Linear Algebra and ML: Teasers

Let's look at a few examples of connections between algebra and ML / CS...

Linear Algebra and ML: Linear Regression

- We want to find $f : \mathbb{R}^d \to \mathbb{R}$ linear (i.e. $f : \mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$) minimizing the squared error loss on the training data $\mathcal{L} = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2$.

Linear Algebra and ML: Linear Regression

- We want to find $f : \mathbb{R}^d \to \mathbb{R}$ linear (i.e. $f : \mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$) minimizing the squared error loss on the training data $\mathcal{L} = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2$.
- Solution: $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ where $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $\mathbf{y} \in \mathbb{R}^N$.
- The vector of prediction is given by $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}^* = \underbrace{\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{\text{orthogonal projection}} \mathbf{y}$

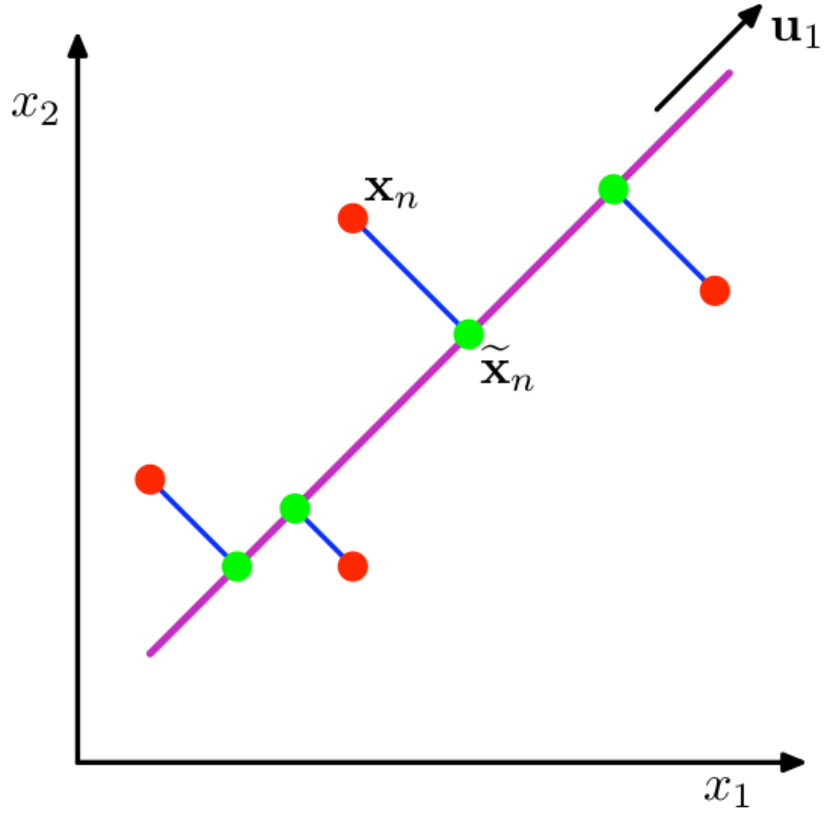
Linear Algebra and ML: Linear Regression

- We want to find $f : \mathbb{R}^d \to \mathbb{R}$ linear (i.e. $f : \mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$) minimizing the squared error loss on the training data $\mathcal{L} = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2$.
- Solution: $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ where $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $\mathbf{y} \in \mathbb{R}^N$.
- The vector of prediction is given by $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}^* = \underbrace{\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{\text{orthogonal projection}} \mathbf{y}$

> $\hat{\mathbf{y}}$ is the orthogonal projection of $\mathbf{y}$ onto the subspace of $\mathbb{R}^N$ spanned by the columns of $\mathbf{X}$.

Linear Algebra and ML: Principal Component Analysis

- Given a set of points $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^d$ (assume centered) we want to find the k-dimensional subspace of $\mathbb{R}^d$ such that he projections of the points onto this subspace
  - have maximal variance
  - stay as close as possible to the original points (in ℓ_2 distance).



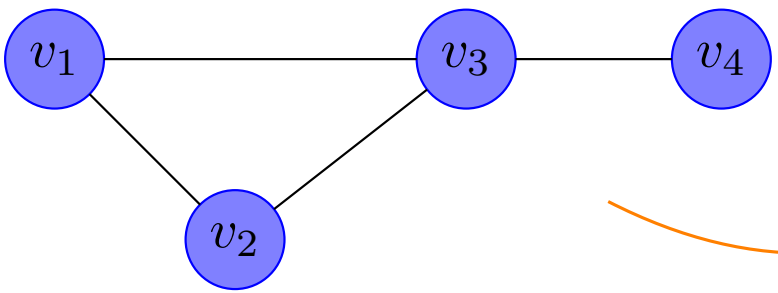
Linear Algebra and ML: Principal Component Analysis

- Given a set of points $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^d$ (assume centered) we want to find the k-dimensional subspace of $\mathbb{R}^d$ such that he projections of the points onto this subspace
  - have maximal variance
  - stay as close as possible to the original points (in ℓ_2 distance).

> The solution is given by the subspace spanned by the top k eigenvectors of the covariance matrix $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$.

Linear Algebra and ML: Spectral Graph Clustering

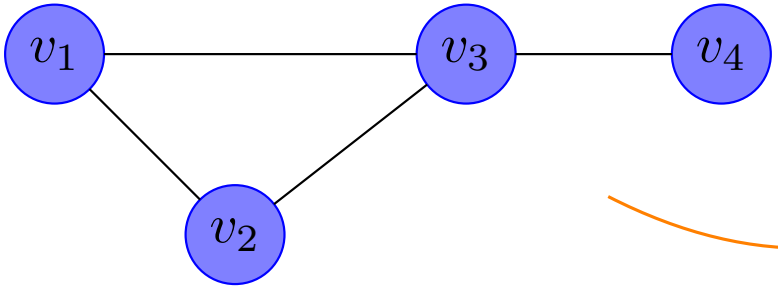
- The Laplacian of a graph is the difference between its degree matrix and its adjacency matrix: $\mathbf{L} = \mathbf{D} - \mathbf{A}$



$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Linear Algebra and ML: Spectral Graph Clustering

- The Laplacian of a graph is the difference between its degree matrix and its adjacency matrix: $\mathbf{L} = \mathbf{D} - \mathbf{A}$



$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Zero is an eigenvalue of the Laplacian, and its multiplicity is equal to the number of connected components of G.

Linear Algebra and ML: Spectral Learning of HMMs/WFAs

- Let Σ be a finite alphabet (e.g. $\Sigma = \{a, b\}$).
- Let Σ^* be the set of all finite sequences of symbols in Σ (e.g. $a, b, ab, aab, bbba, \ldots$).
- Given a real-valued function $f : \Sigma^* \to \mathbb{R}$, its Hankel matrix $\mathbf{H} \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$ is a bi-infinite matrix whose entries are given by $\mathbf{H}_{u,v} = f(uv)$.

Linear Algebra and ML: Spectral Learning of HMMs/WFAs

- Let Σ be a finite alphabet (e.g. $\Sigma = \{a, b\}$).
- Let Σ^* be the set of all finite sequences of symbols in Σ (e.g. $a, b, ab, aab, bbba, \ldots$).
- Given a real-valued function $f : \Sigma^* \to \mathbb{R}$, its Hankel matrix $\mathbf{H} \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$ is a bi-infinite matrix whose entries are given by $\mathbf{H}_{u,v} = f(uv)$.

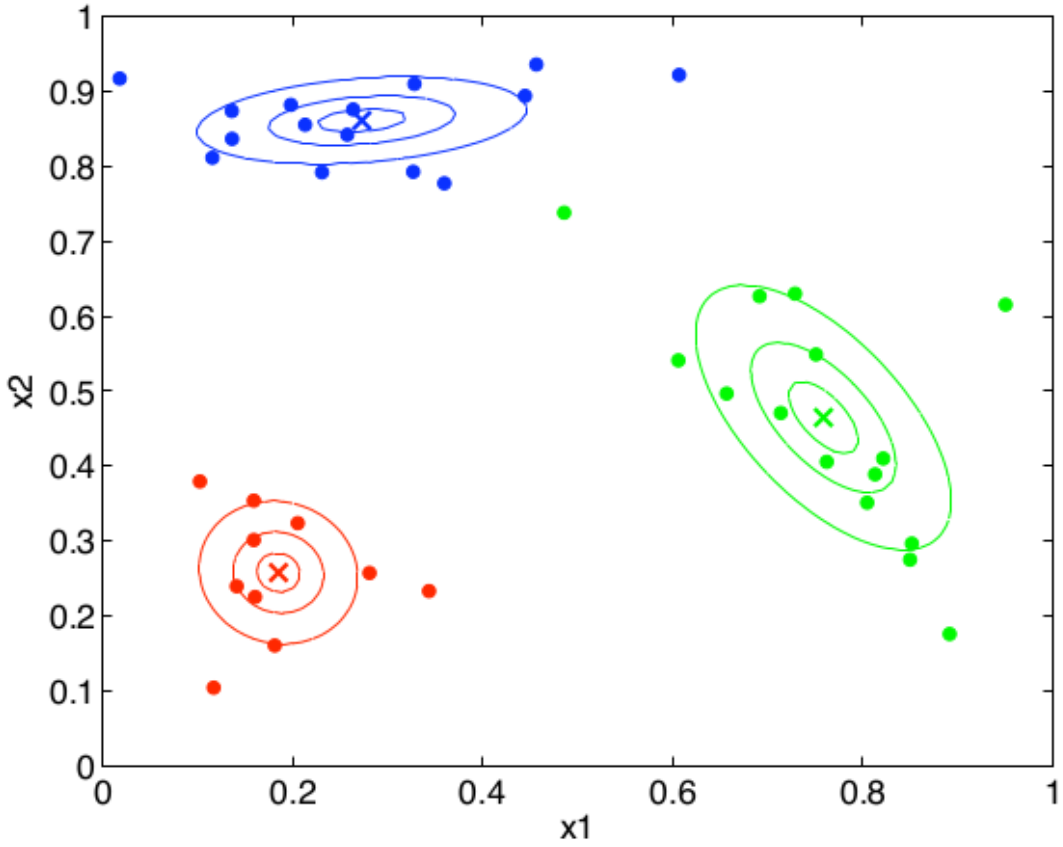
> The rank of $\mathbf{H}$ is finite if and only if f can be computed by a weighted automaton.

(if you don't know what a weighted automaton is, think some kind of RNN with linear activation functions)

Linear Algebra and ML: Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$, i.e. the pdf of $\mathbf{x}$ is

$$f(\mathbf{x}) = \sum_{i=1}^{k} p_i \mathcal{N}(\boldsymbol{\mu}_i, \sigma^2\mathbf{I})$$



Linear Algebra and ML: Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$, i.e. the pdf of $\mathbf{x}$ is

$$f(\mathbf{x}) = \sum_{i=1}^{k} p_i \mathcal{N}(\boldsymbol{\mu}_i, \sigma^2\mathbf{I})$$

> The rank of the (modified) second-order moment
>
> $$\mathbf{M} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \sigma^2\mathbf{I}$$
>
> is at most k.

(we actually have $\mathbf{M} = \sum_i p_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$)

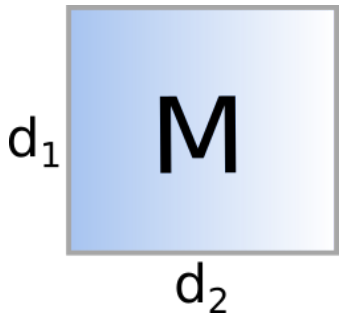
Spectral Methods (high-level view)

- Spectral methods usually achieve learning by extracting structure from observable quantities through eigen-decompositions/tensor decompositions.
- Spectral methods often constitute an alternative to EM to learn latent variable models (e.g. HMMs, single-topic/Gaussian mixtures models).
- Advantages of spectral methods:
  - computationally efficient,
  - consistent,
  - no local optima.

Tensors

What about tensors?

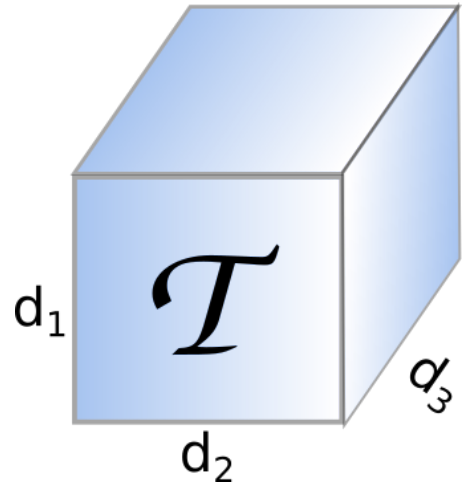
Tensors vs Matrices



$\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$

$\mathbf{M}_{ij} \in \mathbb{R}$ for

$i \in [d_1], j \in [d_2]$

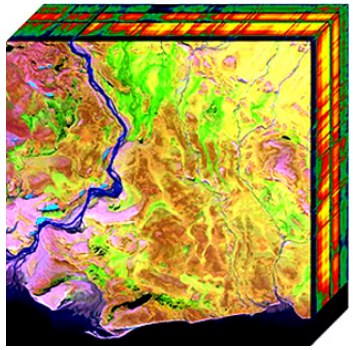


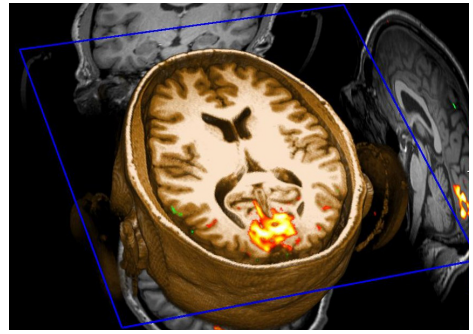
$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

$(\mathcal{T}_{ijk}) \in \mathbb{R}$ for $i \in [d_1], j \in [d_2], k \in [d_3]$

Tensors and Machine Learning

(i) Data has a tensor structure: color image, video, multivariate time series...



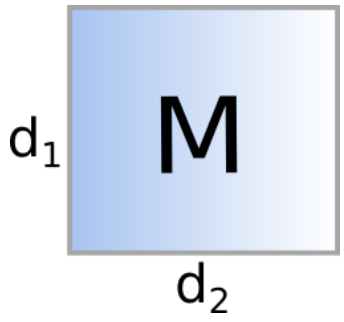


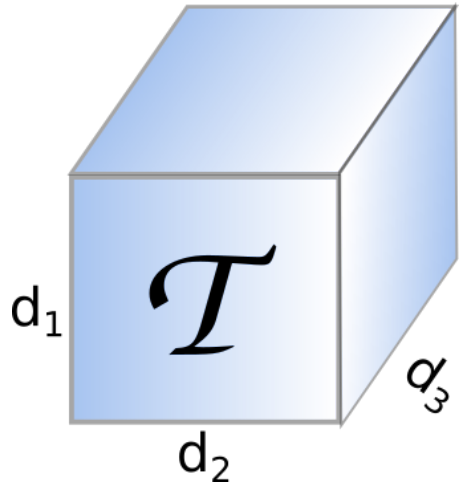
(ii) Tensor as parameters of a model: neural networks, polynomial regression, weighted tree automata...

$$f(\mathbf{x}) = \sum_{i,j,k} \mathcal{T}_{ijk} \mathbf{x}_i \mathbf{x}_j \mathbf{x}_k$$

(iii) Tensors as tools: tensor method of moments, system of polynomial equations, layer compression in neural networks, theoretical analysis of expressiveness of neural networks...

Tensors





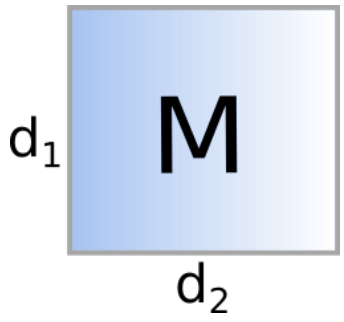
$\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$

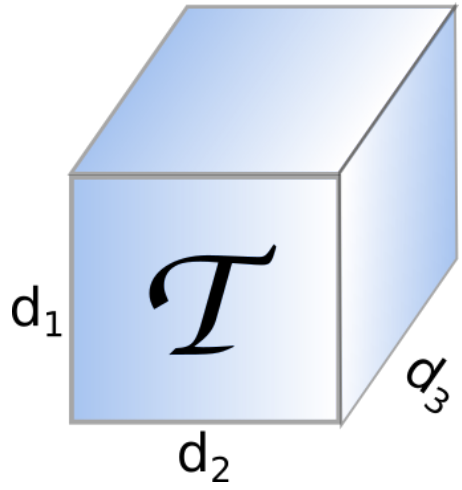
$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

$\mathbf{M}_{ij} \in \mathbb{R}$ for $i \in [d_1], j \in [d_2]$

$(\mathcal{T}_{ijk}) \in \mathbb{R}$ for $i \in [d_1], j \in [d_2], k \in [d_3]$

Tensors





$\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$

$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

$\mathbf{M}_{ij} \in \mathbb{R}$ for $i \in [d_1], j \in [d_2]$ $\quad (\mathcal{T}_{ijk}) \in \mathbb{R}$ for $i \in [d_1], j \in [d_2], k \in [d_3]$

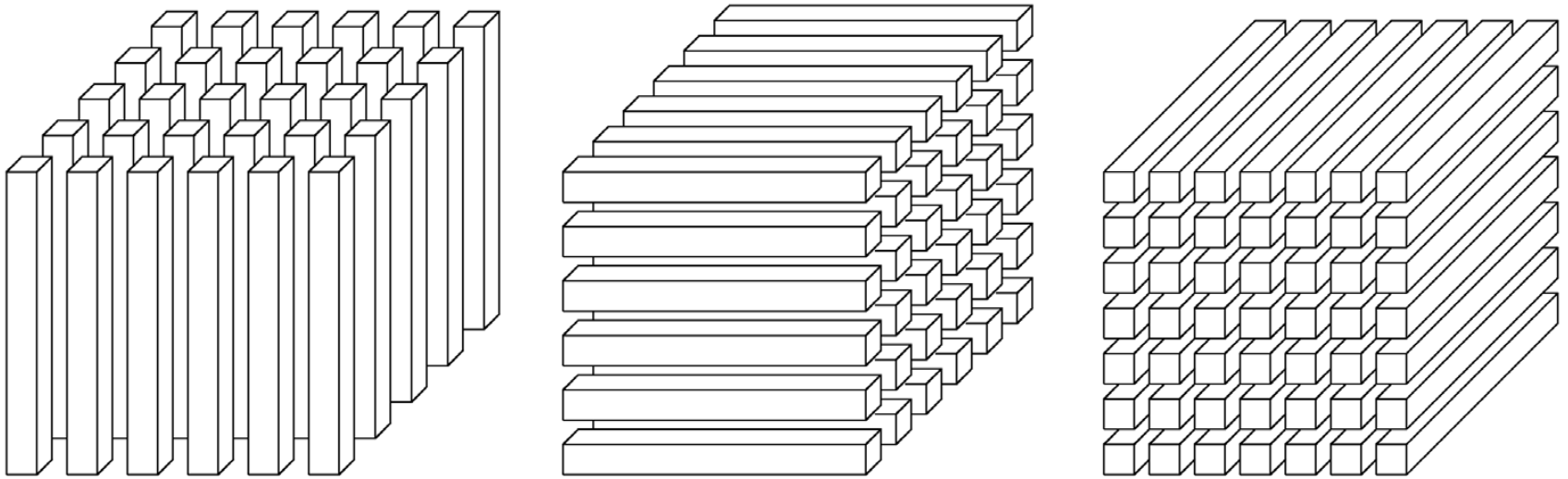
- Outer product. If $\mathbf{u} \in \mathbb{R}^{d_1}$, $\mathbf{v} \in \mathbb{R}^{d_2}$, $\mathbf{w} \in \mathbb{R}^{d_3}$:

$\mathbf{u} \circ \mathbf{v} = \mathbf{u}\mathbf{v}^\top \in \mathbb{R}^{d_1 \times d_2}$ $\qquad (\mathbf{u} \circ \mathbf{v})_{i,j} = \mathbf{u}_i \mathbf{v}_j$

$\mathbf{u} \circ \mathbf{v} \circ \mathbf{w} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ $\qquad (\mathbf{u} \circ \mathbf{v} \circ \mathbf{w})_{i,j,k} = \mathbf{u}_i \mathbf{v}_j \mathbf{w}_k$

Tensors: mode-n fibers

- Matrices have rows and columns, tensors have fibers[1]:



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

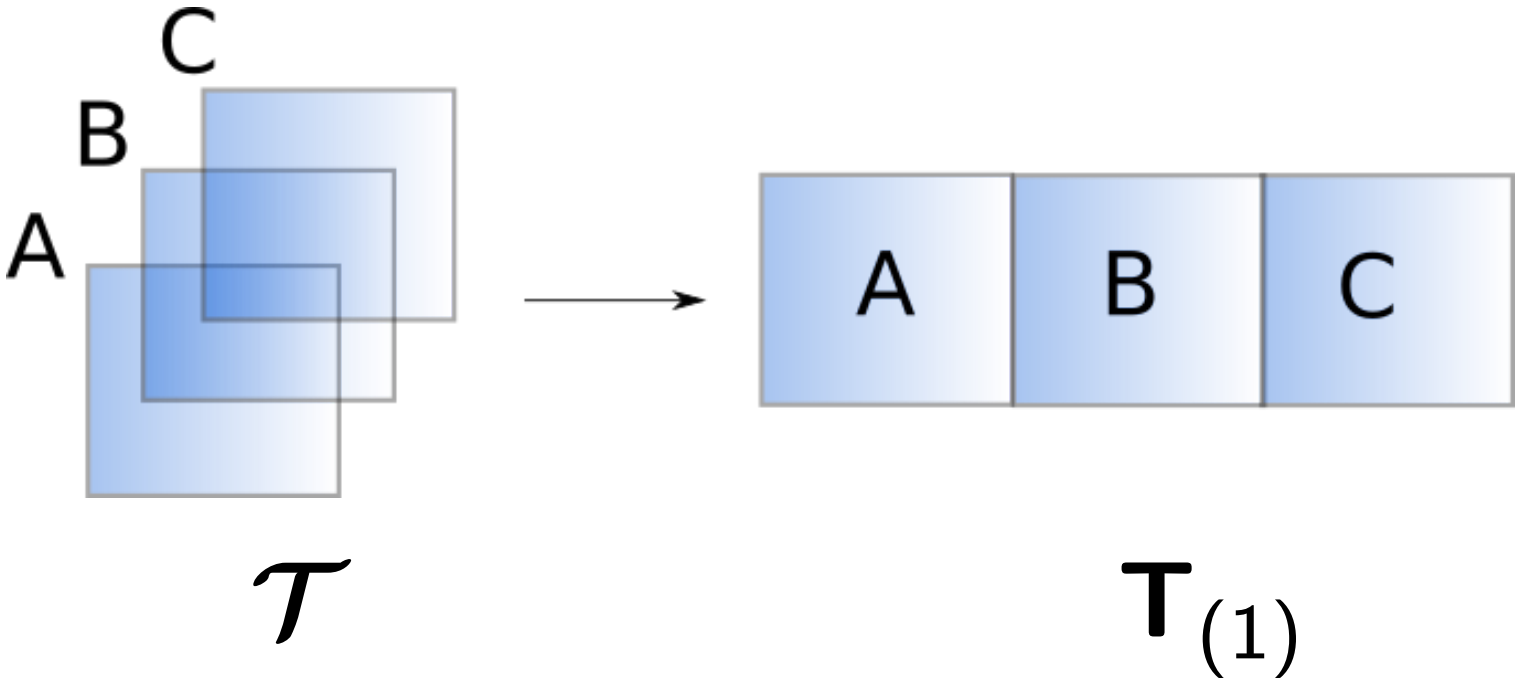
Fig. 2.1 *Fibers of a 3rd-order tensor.*

[1] fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].

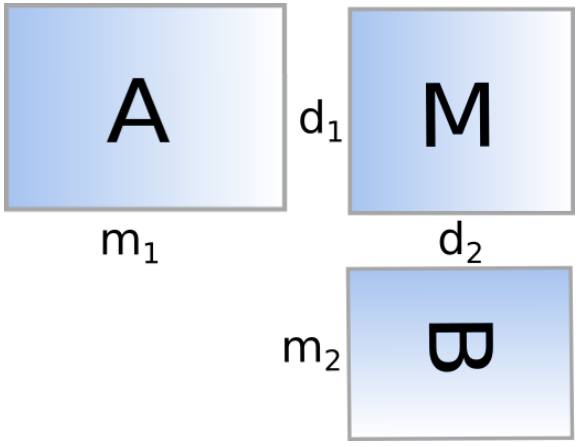
Tensors: Matricizations

- $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ can be reshaped into a matrix as

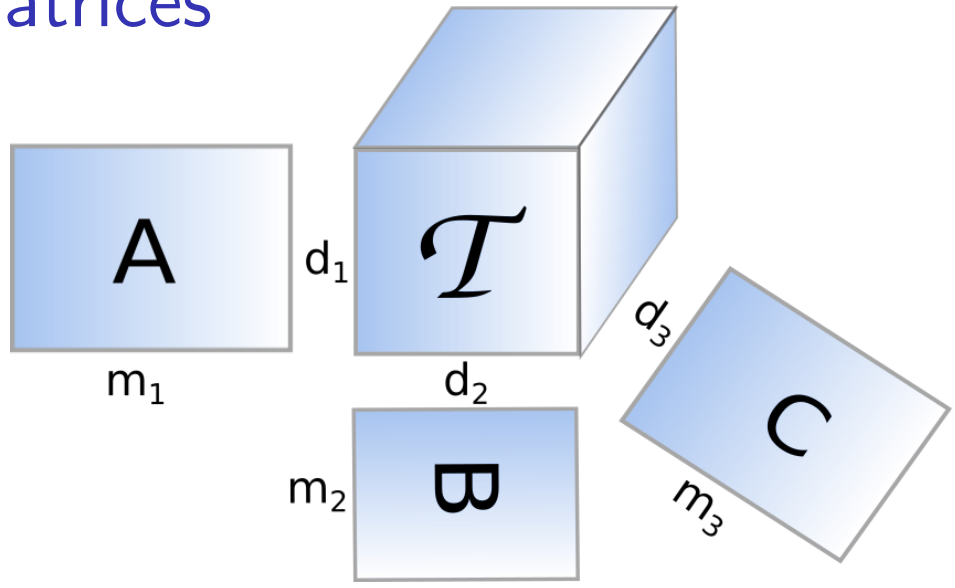
$$\mathbf{T}_{(1)} \in \mathbb{R}^{d_1 \times d_2 d_3}$$
$$\mathbf{T}_{(2)} \in \mathbb{R}^{d_2 \times d_1 d_3}$$
$$\mathbf{T}_{(3)} \in \mathbb{R}^{d_3 \times d_1 d_2}$$



Tensors: Multiplication with Matrices

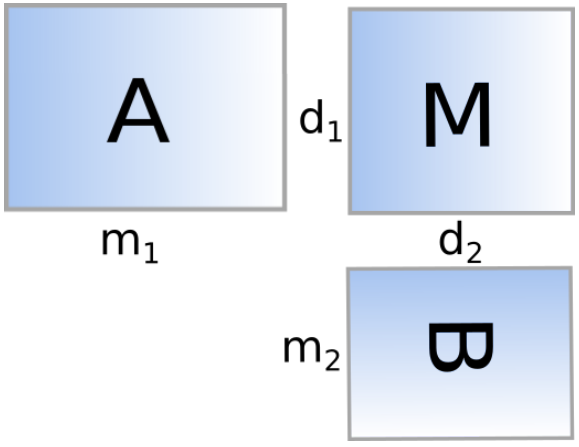


$\mathbf{AMB}^\top \in \mathbb{R}^{m_1 \times m_2}$

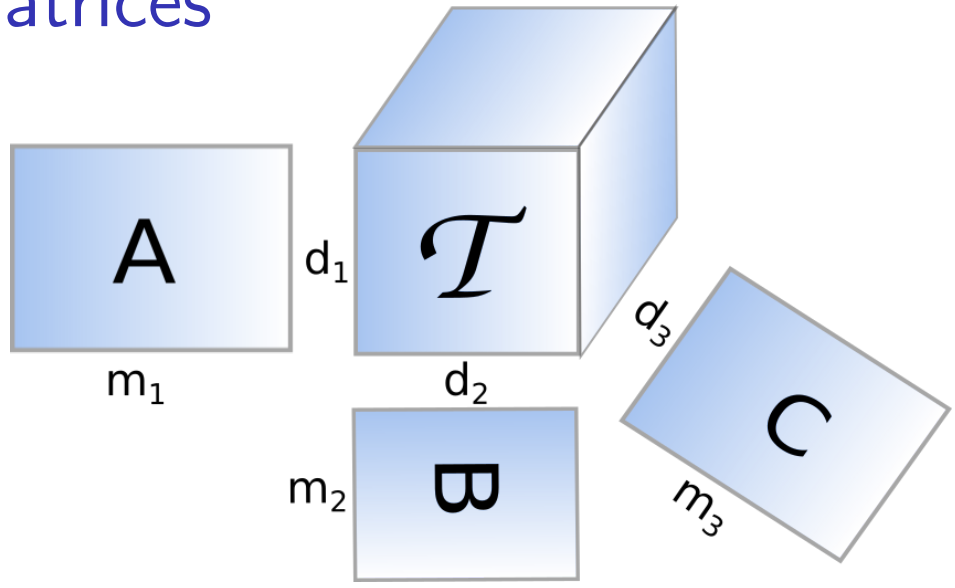


$\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$

Tensors: Multiplication with Matrices



$\mathbf{AMB}^\top \in \mathbb{R}^{m_1 \times m_2}$



$\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$

ex: If $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathbf{B} \in \mathbb{R}^{m_2 \times d_2}$, then $\mathcal{T} \times_2 \mathbf{B} \in \mathbb{R}^{d_1 \times m_2 \times d_3}$ and

$$(\mathcal{T} \times_2 \mathbf{B})_{i_1,i_2,i_3} = \sum_{k=1}^{d_2} \mathcal{T}_{i_1,k,i_3} \mathbf{B}_{i_2,k} \text{ for all } i_1 \in [d_1], i_2 \in [m_2], i_3 \in [d_3].$$

Tensors are not easy...

MOST TENSOR PROBLEMS ARE NP HARD

CHRISTOPHER J. HILLAR AND LEK-HENG LIM

Abstract. The idea that one might extend numerical linear algebra, the collection of matrix computational methods that form the workhorse of scientific and engineering computing, to *numerical multilinear algebra*, an analogous collection of tools involving hypermatrices/tensors, appears very promising and has attracted a lot of attention recently. We examine here the computational tractability of some core problems in numerical multilinear algebra. We show that tensor analogues of several standard problems that are readily computable in the matrix (i.e. 2-tensor) case are NP hard. Our list here includes: determining the feasibility of a system of bilinear equations, determining an eigenvalue, a singular value, or the spectral norm of a 3-tensor, determining a best rank-1 approximation to a 3-tensor, determining the rank of a 3-tensor over $\mathbb{R}$ or $\mathbb{C}$. Hence making tensor computations feasible is likely to be a challenge.

[Hillar and Lim, *Most tensor problems are NP-hard*, Journal of the ACM, 2013.]

Tensors are not easy...

MOST TENSOR PROBLEMS ARE NP HARD

CHRISTOPHER J. HILLAR AND LEK-HENG LIM

Abstract. The idea that one might extend numerical linear algebra, the collection of matrix computational methods that form the workhorse of scientific and engineering computing, to *numerical multilinear algebra*, an analogous collection of tools involving hypermatrices/tensors, appears very promising and has attracted a lot of attention recently. We examine here the computational tractability of some core problems in numerical multilinear algebra. We show that tensor analogues of several standard problems that are readily computable in the matrix (i.e. 2-tensor) case are NP hard. Our list here includes: determining the feasibility of a system of bilinear equations, determining an eigenvalue, a singular value, or the spectral norm of a 3-tensor, determining a best rank-1 approximation to a 3-tensor, determining the rank of a 3-tensor over $\mathbb{R}$ or $\mathbb{C}$. Hence making tensor computations feasible is likely to be a challenge.

[Hillar and Lim, *Most tensor problems are NP-hard*, Journal of the ACM, 2013.]

... but training a neural network with 3 nodes is also NP hard [Blum and Rivest, NIPS '89]

Tensors vs. Matrices: Rank

- The rank of a matrix $\mathbf{M}$ is:
  - the number of linearly independent columns of $\mathbf{M}$
  - the number of linearly independent rows of $\mathbf{M}$
  - the smallest integer R such that $\mathbf{M}$ can be written as a sum of R rank-one matrix:

$$\mathbf{M} = \sum_{i=1}^{R} \mathbf{u}_i \mathbf{v}_i^\top .$$

Tensors vs. Matrices: Rank

- The rank of a matrix $\mathbf{M}$ is:
  - the number of linearly independent columns of $\mathbf{M}$
  - the number of linearly independent rows of $\mathbf{M}$
  - the smallest integer R such that $\mathbf{M}$ can be written as a sum of R rank-one matrix:

$$\mathbf{M} = \sum_{i=1}^{R} \mathbf{u}_i \mathbf{v}_i^\top .$$

- The multilinear rank of a tensor $\mathcal{T}$ is a tuple of integers (R_1, R_2, R_3) where R_n is the number of linearly independent mode-n fibers of $\mathcal{T}$:

$$R_n = \operatorname{rank}(\mathbf{T}_{(n)})$$

Tensors vs. Matrices: Rank

- The rank of a matrix $\mathbf{M}$ is:
  - the number of linearly independent columns of $\mathbf{M}$
  - the number of linearly independent rows of $\mathbf{M}$
  - the smallest integer R such that $\mathbf{M}$ can be written as a sum of R rank-one matrix:

$$\mathbf{M} = \sum_{i=1}^{R} \mathbf{u}_i \mathbf{v}_i^\top.$$

- The multilinear rank of a tensor $\mathcal{T}$ is a tuple of integers (R_1, R_2, R_3) where R_n is the number of linearly independent mode-n fibers of $\mathcal{T}$:

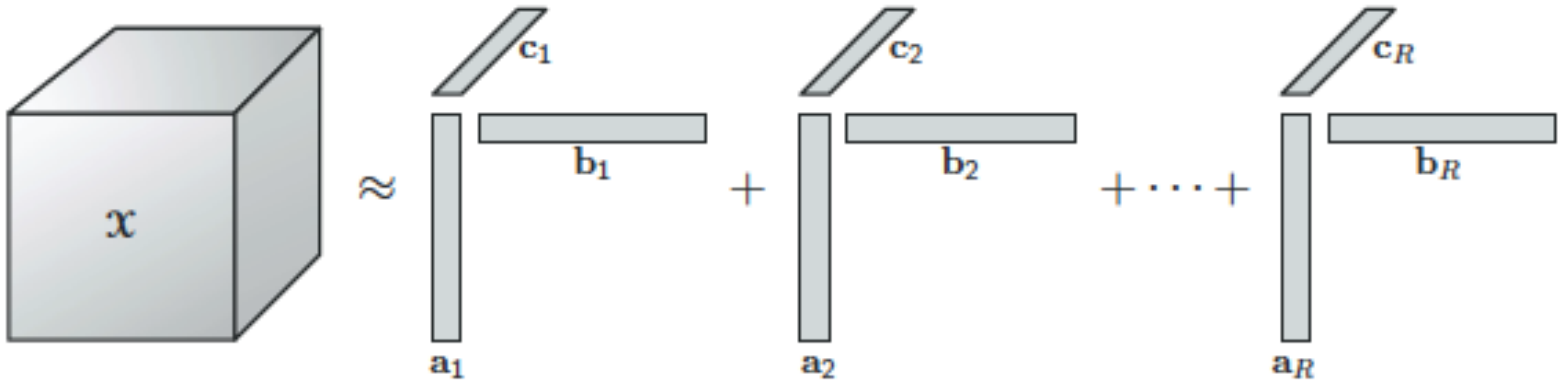
$$R_n = \operatorname{rank}(\mathbf{T}_{(n)})$$

- The CP rank of $\mathcal{T}$ is the smallest integer R such that $\mathcal{T}$ can be written as a sum of R rank-one tensors:

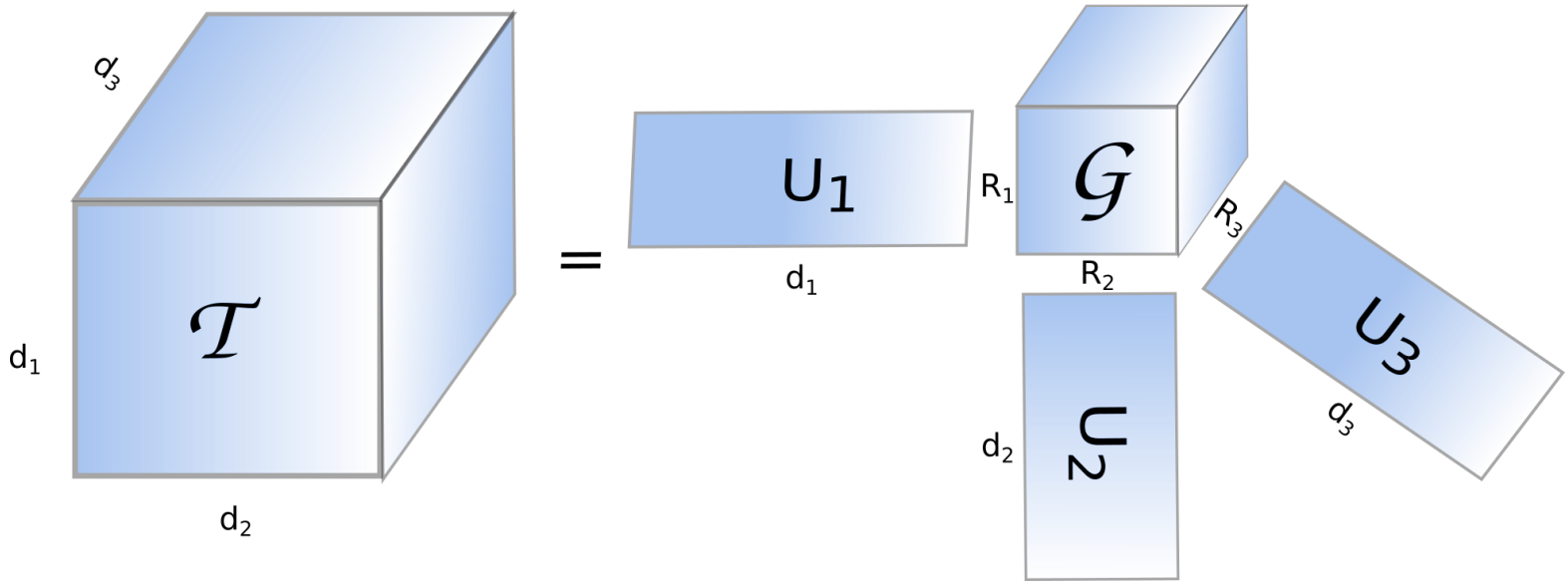
$$\mathcal{T} = \sum_{i=1}^{R} \mathbf{u}_i \circ \mathbf{v}_i \circ \mathbf{w}_i.$$

CP and Tucker decomposition

- CP decomposition[2]:



- Tucker decomposition:



[2]fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].

Hardness results

- Those are all NP-hard for tensor of order ≥ 3 in general:
  - Compute the CP rank of a given tensor
  - Find the best approximation with CP rank R of a given tensor
  - Find the best approximation with multilinear rank $(R_1, \cdots, R_p)$ of a given tensor (*)
  - ...
- On the positive side:
  - Computing the multilinear rank is easy and efficient algorithms exist for (*).
  - Under mild conditions, the CP decomposition is unique (modulo scaling and permutations).

  $\Rightarrow$ Very relevant for model identifiability...

Back to the Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$.

> The (modified) second-order moment $\mathbf{M} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \sigma^2\mathbf{I}$ is such that
>
> $$\mathbf{M} = \sum_{i=1}^{k} p_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$$

- Can we recover the mixing weights p_i and centers $\boldsymbol{\mu}_i$ from $\mathbf{M}$?

Back to the Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$.

> The (modified) second-order moment $\mathbf{M} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \sigma^2\mathbf{I}$ is such that
>
> $$\mathbf{M} = \sum_{i=1}^{k} p_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$$

- Can we recover the mixing weights p_i and centers $\boldsymbol{\mu}_i$ from $\mathbf{M}$?
  - No, except if the $\boldsymbol{\mu}_i$ are orthonormal, in which case they are the eigenvectors of $\mathbf{M}$ and the p_i are the corresponding eigenvalues.

Back to the Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$.

> The (modified) second-order moment $\mathbf{M} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \sigma^2\mathbf{I}$ is such that
>
> $$\mathbf{M} = \sum_{i=1}^{k} p_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$$

- Can we recover the mixing weights p_i and centers $\boldsymbol{\mu}_i$ from $\mathbf{M}$?
  - No, except if the $\boldsymbol{\mu}_i$ are orthonormal, in which case they are the eigenvectors of $\mathbf{M}$ and the p_i are the corresponding eigenvalues.
- But if we know both the matrix $\mathbf{M}$ and the 3rd order tensor $\boldsymbol{\mathcal{T}} = \sum_{i=1}^{k} p_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i$, then we can recover the weights and centers if the $\boldsymbol{\mu}_i$ are linearly independent.
  - ↪ *Anandkumar et al. "Tensor decompositions for learning latent variable models." JMLR 15 (2014): 2773-2832.*